Nonlinear Model Predictive Control of a Cable-Robot-Based Motion Simulator

Katliar, Mikhail; Fischer, Joerg; Frison, Gianluca; Diehl, Moritz; Teufel, Harald; Buelthoff, Heinrich H.

Published in:
I F A C Workshop Series

Link to article, DOI:
10.1016/j.ifacol.2017.08.901

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 9833–9839

Nonlinear Model Predictive Control of a Cable-Robot-Based Motion Simulator

Mikhail Katliar * **Jörg Fischer** ** **Gianluca Frison** ***,** **Moritz Diehl** **** **Harald Teufel** * **Heinrich H. Bülthoff** *

* *Dept. of Human Perception, Cognition and Action, Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tuebingen, Germany (e-mail: [mikhail.katliar, harald.teufel, heinrich.buelthoff]@tuebingen.mpg.de)*
** *Dept. of Microsystems Engineering University of Freiburg, Georges-Koehler-Allee 102, D-79110 Freiburg, Germany (e-mail: [joerg.fischer]@imtek.uni-freiburg.de)*
*** *Technical University of Denmark, (e-mail: [giaf]@dtu.dk)*
**** *Dept. of Microsystems Engineering & Dept. of Mathematics University of Freiburg, Georges-Koehler-Allee 102, D-79110 Freiburg, Germany (e-mail: moritz.diehl@imtek.uni-freiburg.de)*

Abstract: In this paper we present the implementation of a model-predictive controller (MPC) for real-time control of a cable-robot-based motion simulator. The controller computes control inputs such that a desired acceleration and angular velocity at a defined point in simulator's cabin are tracked while satisfying constraints imposed by working space and allowed cable forces of the robot. In order to fully use the simulator capabilities, we propose an approach that includes the motion platform actuation in the MPC model. The tracking performance and computation time of the algorithm are investigated in computer simulations. Furthermore, for motion simulation scenarios where the reference trajectories are not known beforehand, we derive an estimate on how much motion simulation fidelity can maximally be improved by any reference prediction scheme compared to the case when no prediction scheme is applied.

Keywords: model predictive control, cable robots, vehicle simulators, motion cueing.

1. INTRODUCTION

Motion simulators are devices for creating an experience of being inside a moving vehicle. They are widely used for training of aircraft pilots and racing car drivers, as well as for studying human motion perception and control behavior. A motion simulator typically consists of a motion system (moving platform) and a visualization system. The human vestibular system senses inertial forces and angular velocity. We refer to these quantities, as functions of time, as the "inertial signal". A control algorithm that drives the moving platform to reproduce a certain inertial signal is called *motion cueing algorithm* (MCA). Different types of MCAs are reviewed in Garrett and Best (2010). Although the knowledge about human vestibular system is often used in MCA design (Telban and Cardullo, 2005), in this paper we do not consider perceptual aspects and focus on reproduction of the physical stimulus (inertial signal).

The major difficulty concerning motion cueing is that the quantities that need to be reproduced (velocities and accelerations) are 1st and 2nd order derivatives of the coordinates. If reproduced one-to-one and integrated over time, this can easily result in coordinates outside the motion system's physical limits. Therefore, the inertial signal in a simulator has to be necessarily distorted, in order to keep the resulting trajectory within the limits of the motion system.

The classical way to deal with this problem (Schmidt and Conrad, 1970; Nahon and Reid, 1990) is to split the acceleration signal into a low-frequency and a high-frequency part. The high-frequency part is reproduced directly by accelerating the motion platform, while the low-frequency part is reproduced by tilting the platform so that the changing direction of the gravity creates an acceleration which matches the desired one (tilt-coordination). This, however, creates a rotation which is not present in the original signal (false cue). To ensure that the motion system stays within its limits for any anticipated inertial signal, the reference signal is usually scaled down. As a consequence, the reproduced forces are reduced and the realism of the simulation decreases.

Sivan et al. (1982) proposed a solution based on optimal LQR-control and Nahon et al. (1992) proposed a variation of the classical algorithm employing adaptive filters. However, all these solutions require parameter tuning to ensure that the constraints of the motion system are respected.

Recently, Dagdelen et al. (2009) proposed to use model predictive control (MPC) for motion simulation. In MPC, control inputs are calculated at each time step by numerically minimizing a cost function that reflects (among

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2017.08.901

others) the deviation of the predicted output trajectory from the desired reference trajectory over N time steps (Rawlings and Mayne, 2009), where N is called the *horizon length*. A huge advantage of MPC is that it can directly consider input, state, and output constraints of the plant in the calculation of the control inputs.

In theory, MPC results in a close to optimal tracking performance, but comes at the cost of computational complexity and implementation effort. The main parameter affecting the trade-off between tracking performance and computation time is the MPC horizon length N: the bigger N, the better the tracking performance and the longer it takes to compute the control input.

In a scheme proposed by Dagdelen et al. (2009), an MCA consists of two parts: an MPC-based feedforward compensator that defines the trajectory to follow, and a feedback controller that handles actuator dynamics. This makes the dynamics of the motion system linear and decouples them from the dynamics of the actuators. The same approach is adopted by Fang and Kemeny (2012); Beghi et al. (2012). However, because the actuators can have different capabilities depending of the state of the system, in such implementations, an MPC controller must be conservative about the actuation system capabilities, so that the trajectory generated by the MPC can always be realized by the actuators. We propose an MPC formulation that includes the actuation of the motion platform. By doing so, the MPC controller is able to directly handle the constraints of the actuation system, making full use of simulator capabilities.

Integrating the actuation dynamics of the platform into the MPC model increases the dimension of the state space, and thus increases the size of the optimization problem solved by the MPC and the required computational effort. Together with long horizon lengths typically required for motion simulation applications (Katliar et al., 2015), it makes the real-time implementation of the proposed approach challenging.

In this paper, we derive a real-time capable implementation of a model predictive controller for a cable-robot-based motion simulator that includes the actuation dynamics, and investigate the trade-off between tracking and computational performance for the derived controller in simulation. In this context, we also derive an estimate of how much motion simulation fidelity can maximally be improved by any reference prediction scheme if the reference trajectories for the motion simulation are not known beforehand.

2. THE MPI CABLE ROBOT SIMULATOR

The Cable Robot Simulator (CRS) at the Max Planck Institute for Biological Cybernetics in Tuebingen, Germany, is used for human motion perception research and virtual reality experiments. It consists of a mobile platform which is connected to 8 fixed winches by cables passing through redirection pulleys (Fig. 1). The winch motors can be controlled independently in velocity- or torque-control mode. The motors are equipped with encoders providing angular positions of the rotors. The cable forces are measured by sensors located in the redirection pulleys. The mobile platform is equipped with an inertial measurement unit (IMU) that provides real-time measurements of accelerations and angular velocities.

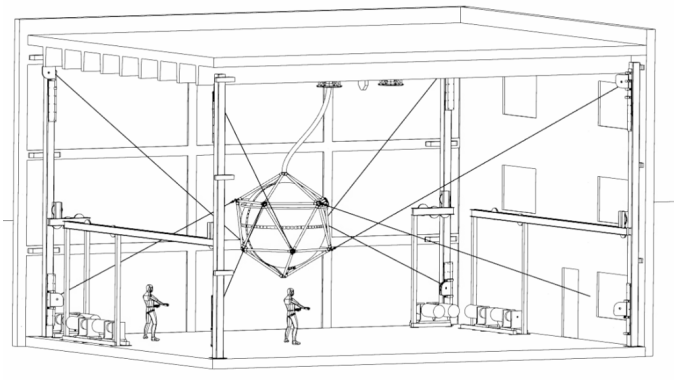

Fig. 1. The MPI Cable Robot Simulator. *Image credit: Philipp Miermeister, MPI for Biological Cybernetics.*

Max. acceleration [linear, angular]	$[14\,\mathrm{m/s^2},\ 100\,^\circ/\mathrm{s^2}]$
Max. velocity [linear, angular]	$[5\,\mathrm{m\,s^{-1}},\ 100\,^\circ\,\mathrm{s^{-1}}]$
Safe translational workspace [X, Y, Z]	[4 m, 5 m, 5 m]
Rotational workspace [roll, pitch, yaw]	$[\pm 40^\circ,\ \pm 40^\circ,\ \pm 5^\circ]$
Feasible cable tension [min, max]	[1000 N, 14 000 N]
Max payload	500 kg

Table 1. Physical constraints of the CRS (from Miermeister et al. (2016)).

The main constraints of the system dictated by its design and safety requirements are summarized in Table 1. A detailed description of the setup can be found in (Miermeister et al., 2016).

3. THE MODEL PREDICTIVE MOTION CUEING CONTROLLER

In this section, an MPC controller is derived that calculates optimal cable forces to be realized by the winch motors of the cable robot to track a given inertial signal inside the cabin under satisfaction of all constraints of the robot. To this end, first a nonlinear dynamical model of the cable robot is derived (Sec. 3.1). Based on this, the control task is formulated as a nonlinear receding horizon optimization problem (Sec. 3.2). Finally, the implementation of a real-time capable model predictive controller is presented that numerically solves this optimization problem at each time sample (Sec 3.3).

3.1 Dynamical model of the CRS

We consider the motion platform to be a rigid body with a known mass m and inertia tensor I. The mass and elasticity of the cables, the friction forces, the dynamics of the motors, winches and pulleys are not included in our model. The origin of the platform reference frame (PF) is placed in the center of mass. Let $\mathbf{r}$ be the position of the platform's center of mass in world coordinate frame (WF), $\mathbf{q}$ be the quaternion that defines the rotation from PF to WF, $\mathbf{v}$ be the velocity of the center of mass in WF, and ω be the angular velocity of the platform in PF. The state of the system is

$$\mathbf{x} = [\mathbf{r}^\top, \mathbf{q}^\top, \mathbf{v}^\top, \omega^\top]^\top \in \mathbb{R}^{n_x},\ n_x = 13\ . \qquad (1)$$

The platform is controlled by changing cable tension forces. We define number of cables as N_c, coordinates of outlet point of i-th cable in WF as $\mathbf{a}_i$, coordinates of anchor point of i-th cable on the mobile platform in PF as $\mathbf{b}_i$, and let $\mathbf{u} = [u_1, u_2, \ldots, u_{N_c}]^\top$ be the vector of cable tension forces. The force acting on the platform from i-th cable, expressed in WF, is

$$\mathbf{F}_i = \frac{\mathbf{l}_i}{\|\mathbf{l}_i\|} u_i \ ,$$

where $\mathbf{l}_i = \mathbf{a}_i - \mathbf{r} - R(\mathbf{q})\mathbf{b}_i$ is the vector connecting anchor point of i-th cable with its outlet point (in WF), and $R(\mathbf{q})$ is the rotation matrix from PF to WF corresponding to quaternion $\mathbf{q}$. Based on rigid-body dynamics, following ODEs are derived that describe the dynamics of the system:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{q}} &= \frac{1}{2} G(\mathbf{q})^\top \omega \\ \dot{\mathbf{v}} &= \frac{1}{m} \sum_{i=1}^{N_c} \mathbf{F}_i + \mathbf{g} \\ \dot{\omega} &= I^{-1} \left(\sum_{i=1}^{N_c} \mathbf{b}_i \times \left(R(\mathbf{q})^\top \mathbf{F}_i \right) - \omega \times (I\omega) \right) \ , \end{aligned} \tag{2}$$

where $G(\mathbf{q})$ is given by

$$G(\mathbf{q}) = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \ .$$

Let $F(\mathbf{x}_0, \mathbf{u}_0, t)$ be the solution of (2) at time t given initial condition $\mathbf{x}(0) = \mathbf{x}_0$ and $\mathbf{u}(\tau) = \mathbf{u}_0 \ \forall \tau \in [0, t)$. We discretize the model by assuming piecewise-constant cable force inputs:

$$\begin{aligned} \mathbf{u}(t) = \mathbf{u}_k \in \mathbb{R}^{n_u}, \quad n_u = N_c = 8 \\ kT \le t < (k+1)T, \quad k = 0, 1, 2, \ldots, \end{aligned} \tag{3}$$

where T is the sampling time. We denote the system state at sampling point $t = kT$ as

$$\mathbf{x}_k = \mathbf{x}(kT) \ .$$

Then, the ODEs (2) can be converted to the discrete-time dynamic equations via direct multiple shooting:

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k, T), \ k = 0, 1, 2, \ldots \ .$$

3.2 Formulation of the optimal control problem

In a motion simulator, we want to reproduce inertial forces and angular velocities in the cabin as close as possible to a given reference profile. We express these quantities as functions of system state and control input. Consider a point mass m attached to the platform at the origin of PF. $-\mathbf{F}$ denotes the platform reaction force acting on the mass. According to Newton's 2nd law,

$$m\dot{\mathbf{v}} = R(\mathbf{q})(-\mathbf{F}) + m\mathbf{g} \ , \tag{4}$$

where $-\mathbf{F}$ is expressed in PF. The quantity

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) := \frac{\mathbf{F}}{m} = R(\mathbf{q})^\top (\mathbf{g} - \dot{\mathbf{v}}) \tag{5}$$

is the so called *specific force*, which is measured in $\mathrm{m/s^2}$ and is essentially the "inertial force" in PF divided by mass. It is the combined result of gravity and accelerated motion of the platform.

The angular velocity of PF relative to WF, expressed in PF, is simply $\omega = \omega(\mathbf{x})$. The total inertial signal produced by the platform is

$$\mathbf{y}(\mathbf{x}, \mathbf{u}) = [\mathbf{f}(\mathbf{x}, \mathbf{u})^\top, \omega(\mathbf{x})^\top]^\top \ \in \mathbb{R}^{n_y}, \ n_y = 6 \ . \tag{6}$$

Let $\tilde{\mathbf{x}}_0$ be the state of the system at current time t_0 and $\hat{\mathbf{y}}_k$ be the reference inertial signal at time $t_0 + kT$. At every time sample, the MPC controller solves the following nonlinear constrained minimization problem:

$$\begin{aligned} \underset{\substack{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1} \\ \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N}}{\text{minimize}} \quad & \frac{1}{N} \sum_{k=0}^{N-1} l_k(\mathbf{x}_k, \mathbf{u}_k) + l_N(\mathbf{x}_N) \\ \text{subject to} \quad & \mathbf{x}_0 = \tilde{\mathbf{x}}_0, \\ & \mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k, T), \ k = 0 \ldots N-1, \\ & \mathbf{x}_{\min} \le \mathbf{x}_k \le \mathbf{x}_{\max}, \quad k = 1 \ldots N-1, \\ & \mathbf{x}_{\min,N} \le \mathbf{x}_N \le \mathbf{x}_{\max,N}, \\ & \mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max}, \quad k = 0 \ldots N-1 \end{aligned} \tag{7}$$

where

$$\begin{aligned} l_k(\mathbf{x}_k, \mathbf{u}_k) = & \left\| \mathbf{y}(\mathbf{x}_k, \mathbf{u}_k) - \hat{\mathbf{y}}_k \right\|^2_{W_\mathbf{y}} \\ & + \|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2_{W_\mathbf{x}} + \|\mathbf{u}_k - \hat{\mathbf{u}}_k\|^2_{W_\mathbf{u}} \ , \end{aligned} \tag{8}$$

$$l_N(\mathbf{x}_N) = \|\mathbf{x}_N - \hat{\mathbf{x}}_N\|^2_{W_{\mathbf{x}_N}} \ . \tag{9}$$

The first term in (8) corresponds to reference signal tracking, and $W_\mathbf{y}$ defines weighting of different components of the inertial signal. The second and the third terms allow the controller to track a given state trajectory $\hat{\mathbf{x}}_k$ and input trajectory $\hat{\mathbf{u}}_k$, and $W_\mathbf{x}$, $W_{\mathbf{x}_N}$, $W_\mathbf{u}$ are symmetric positive-definite weighting matrices. The first term is the most important for motion simulation applications. However, by changing the weighting matrices $W_\mathbf{y}$, $W_\mathbf{x}$, $W_\mathbf{u}$, we can balance between inertial signal tracking, state tracking and input tracking, without restarting the controller. This can be necessary, for example, to move the platform to a desired position, etc.. Furthermore, since the system is overactuated, the third term is important to guarantee local convexity of the problem (7). The position and velocity constraints of the system are included in $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$, whereas $\mathbf{u}_{\min}$, $\mathbf{u}_{\max}$ correspond to minimum and maximum feasible cable forces. The bounds $\mathbf{x}_{\min,N}$, $\mathbf{x}_{\max,N}$ of the terminal state $\mathbf{x}_N$ are chosen to constrain the linear and angular velocity of the cabin to small values. This ensures that the motion can be stopped at the end of the control horizon.

3.3 Implementation of the MPC controller

The Real Time Iteration (RTI) scheme (Diehl, 2001) is a Sequential Quadratic Programming (SQP) scheme that performs a single Newton iteration per sampling interval. The rationale behind this is that it is better to return an approximate solution as soon as possible, than iterating until a high-accuracy solution is found while the system is changing and the solution is getting outdated. The RTI scheme is divided into a preparation step (performing all tasks that can be performed before the latest state estimate is available, such as model integration, sensitivity generation and linearization of the objective and constraints) and a feedback step (comprising initial value embedding and Quadratic Program (QP) solution).

At each SQP iteration a QP is solved by a so called *QP solver* algorithm. A variety of QP solvers with different features exist; a good review can be found in Kouzoupis et al. (2015). We initially considered `qpOASES` (Ferreau et al., 2014), `qpDUNES` (Frasch et al., 2013), `HPMPC` (Frison et al., 2014) and `FORCES` (Domahidi et al., 2012) as solvers to be used in our MPC controller. These solvers are tailored for MPC applications, have `C` interface and a free license. However, we excluded FORCES because it requires re-generating and re-compiling the code whenever the horizon length is changed, which is not feasible within our software architecture. We also could not use `qpDUNES` because of software-related issues which we could not solve even with the help of `qpDUNES` developers. Therefore, we could select either `qpOASES` or `HPMPC` to be used by the controller. The features of the two solvers are outlined below.

`qpOASES` implements a general purpose Active Set (AS) method, and it assumes the Hessian matrix to be dense. Therefore, it is combined with a condensing algorithm, that rewrites the large structured KKT matrix of the MPC problem into a small dense one by removing the states from the problem formulation. The implemented condensing algorithm requires $\mathcal{O}(N^3)$ flops. The Cholesky factorization of the dense Hessian matrix requires $\frac{1}{3}(Nn_u)^3$ flops, and therefore in our implementation this approach scales cubically with the horizon length N. We refer to the combination of the implemented condensing algorithm and the `qpOASES` algorithm as `qpOASES+C`. We perform condensing during the feedback phase, therefore the feedback phase time includes execution time of both condensing algorithm and `qpOASES`.

`HPMPC` is a Mehrotra's type Interior Point (IP) method tailored to MPC problems. It employs a Riccati recursion to efficiently factorize the KKT matrix of the unconstrained MPC sub-problems in the computation of the search direction. The Riccati recursion is the most computationally expensive operation at each IP iteration, and it requires $\mathcal{O}(N(n_x + n_u)^3)$ flops. It makes use of linear algebra routines specially tailored to get high-performance also for small matrices. Therefore, this approach scales linearly in N and cubically in $n_x + n_u$.

Generally speaking, AS methods can be easily and effectively warm started, but in case of a bad initial guess they may require many iterations to converge. Conversely, IP methods cannot exploit information about the solution at the previous sampling time, but they are known to generally converge in a rather small number of iterations regardless of the problem instance. We compare the performance of the two solvers in Sec. 4.

The `CasADi` framework (Andersson, 2013) was used to generate C code for the ODEs (2), the output function (6), and the corresponding sensitivities. The differential equations (2) are integrated using one step of explicit 4-th order Runge-Kutta method. The Runge-Kutta integrator and the rest of the controller code was written in C++ and linked to `qpOASES` and `HPMPC` libraries. The reference signal is externally provided to the controller at each sampling time and is defined by a $n_y \times (N+1)$ matrix $[\hat{\mathbf{y}}_0, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_N]$.

Sampling time T	$50\,\mathrm{ms}$
Platform mass m	$190\,\mathrm{kg}$
Inertia tensor I	$\begin{bmatrix} 82.88 & -6.42 & 1.33 \\ -6.42 & 82.49 & 2.20 \\ 1.33 & 2.20 & 94.98 \end{bmatrix}\,\mathrm{kg\,m^2}$
For $k = 0, 1, \ldots, N$:	
$\mathbf{x}_{\min,k}$	$[\mathbf{r}_{\min,k}^\top, -\infty_{1\times 4}, \mathbf{v}_{\min,k}^\top, \omega_{\min,k}^\top]^\top$
$\mathbf{x}_{\max,k}$	$[\mathbf{r}_{\max,k}^\top, \infty_{1\times 4}, \mathbf{v}_{\max,k}^\top, \omega_{\max,k}^\top]^\top$
$\hat{\mathbf{x}}_k$	$[0, 0, 2.5\,\mathrm{m}, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]^\top$
For $k = 0, 1, \ldots, N-1$:	
$\mathbf{r}_{\min,k}$	$[-2, -2, 1]^\top\,\mathrm{m}$
$\mathbf{r}_{\max,k}$	$[2, 2, 4.]^\top\,\mathrm{m}$
$\mathbf{v}_{\min,k}$	$-[5, 5, 5]^\top\,\mathrm{m\,s^{-1}}$
$\mathbf{v}_{\max,k}$	$[5, 5, 5]^\top\,\mathrm{m\,s^{-1}}$
$\omega_{\min,k}$	$-[100, 100, 100]^\top\,^\circ\,\mathrm{s^{-1}}$
$\omega_{\max,k}$	$[100, 100, 100]^\top\,^\circ\,\mathrm{s^{-1}}$
$\mathbf{u}_{\min,k}$	$[1, 1, 1, 1, 1, 1, 1, 1]^\top\mathrm{kN}$
$\mathbf{u}_{\max,k}$	$[9, 9, 9, 9, 9, 9, 9, 9]^\top\mathrm{kN}$
$\hat{\mathbf{u}}_k$	$[5, 5, 5, 5, 5, 5, 5, 5]^\top\mathrm{kN}$
$W_{\mathbf{x},k}$	$0.001 \cdot \mathbb{1}_{n_x}$
Terminal state bounds	
$\mathbf{v}_{\min,N}$	$-[0.01, 0.01, 0.01]^\top\,\mathrm{m\,s^{-1}}$
$\mathbf{v}_{\max,N}$	$[0.01, 0.01, 0.01]^\top\,\mathrm{m\,s^{-1}}$
$\omega_{\min,N}$	$-[0.01, 0.01, 0.01]^\top\,\mathrm{rad\,s^{-1}}$
$\omega_{\max,N}$	$[0.01, 0.01, 0.01]^\top\,\mathrm{rad\,s^{-1}}$
Weighting matrices	
$W_{\mathbf{x}_N}$	$0.001 \cdot \mathbb{1}_{n_x}$
$W_{\mathbf{y}}$	$\mathrm{diag}([1, 1, 1, 100, 100, 100])$
$W_{\mathbf{u}}$	$\mathrm{diag}([0.01, 0.01, 0.01, 0.01, 0.01, 0.01])$

Table 2. Parameters used in the simulation.

The parameter values used in the simulation are summarized in Table 2. The $W_{\mathbf{y}}$ matrix is chosen to approximately equalize the variance of specific force in $\mathrm{m/s^2}$ and angular velocity in $\mathrm{rad\,s^{-1}}$. The reference state $\hat{\mathbf{x}}_k$ is the same for all $k = 0, 1, \ldots, N$ and corresponds to the "neutral" position of the cabin in the middle of the workspace and with the upright orientation. The values $\hat{\mathbf{u}}_k = \hat{\mathbf{u}}$ are chosen to be equal for all $k = 0, 1, \ldots, N-1$ where $\hat{\mathbf{u}}$ is set to prefer cable force values in the middle of the range.

4. PERFORMANCE ASSESMENT

4.1 Tracking performance

The primary objective of the motion simulator is to reproduce a desired reference inertial signal. As a measure of the reference tracking performance, we use the root mean square (RMS) of the following quantities:

$$\begin{aligned} e_{\mathbf{f}} &= \|\mathbf{f} - \hat{\mathbf{f}}\|_2 && \text{(specific force error)}, \\ e_{\omega} &= \|\omega - \hat{\omega}\|_2 && \text{(angular velocity error)}, \\ e_{\mathbf{y}} &= \|\mathbf{y} - \hat{\mathbf{y}}\|_{W_{\mathbf{y}}} && \text{(total error)}. \end{aligned} \tag{10}$$

At each time step the MPC controller needs the reference signal $\hat{\mathbf{y}}_k$, $k = 0, 1, \ldots, N$. However, when a human operator actively controls the simulated vehicle, the future inertial signal is not known, because the driver control action is uncertain. To predict the reference, different strategies can be employed. Denoting the absolute time at the beginning of each control cycle as t_0, we consider the two following extreme cases of reference prediction:

(1) The "constant" strategy sets $\hat{\mathbf{y}}_k = \hat{\mathbf{y}}_{\mathrm{true}}(t_0)\ \forall k = 0 \ldots N$. It corresponds to the case when nothing is known about the future behavior of $\hat{\mathbf{y}}(t)$.

(2) The "oracle" strategy sets $\hat{\mathbf{y}}_k = \hat{\mathbf{y}}_{\text{true}}(t_0 + kT)\ \forall k = 0 \dots N$ when the future reference signal is exactly known for $t \in [t_0, t_0 + NT)$.

The "constant" strategy is the simplest to implement since it does not require any knowledge about the driver and vehicle behavior. On the other hand, the best tracking performance can be achieved in the case when the future inertial signal is known exactly, which occurs in practice when the person is not controlling the simulated vehicle and a pre-defined motion is played back. For the human-in-the-loop scenario, a reasonable and practically possible prediction strategy will result in a tracking performance somewhere between these two extremes. Comparing the two extreme cases gives us an idea of the upper bound of the improvement that can be made by employing a reference prediction strategy.

4.2 Computational performance

In MPC, the calculation of the control input must be performed within a single time step. For horizon length N, the MPC computation time $T_{\text{mpc}}(N)$ is the sum of RTI preparation phase time $T_{\text{p}}(N)$ and RTI feedback phase time $T_{\text{f}}(N)$. In general, the computation time may vary for different input data, depending on the algorithm. Therefore, it is the maximum (across all possible input data) computation time that limits realtime-feasibility of any MPC implementation, which can be estimated as

$$\max T_{\text{mpc}}(N) = \max T_{\text{p}}(N) + \max T_{\text{f}}(N)\ . \qquad (11)$$

In order to restrict maximum computation time, a limit $\overline{N}_{\text{iter}}$ can be set on the number N_{iter} of interior-point method iterations in `HPMPC`. The maximum MPC computation time for the `HPMPC`-based implementation can be then estimated as

$$\max T_{\text{mpc}}(N) = \max T_{\text{p}}(N) + \overline{N}_{\text{iter}} \max T_{\text{iter}}(N)\ , \quad (12)$$

where $T_{\text{iter}}(N)$ is the time of one interior-point method iteration in `HPMPC`.

4.3 Method and dataset

To assess reference tracking performance and computational effort of the MPC controller, we perform simulations on a set of 12 reference inertial signals, corresponding to typical car and helicopter maneuvers. Most of the reference signals were recorded in real vehicles using the `VIASync` recorder (Venrooij, 2014), which provides a synchronized recording of visual, inertial and auditory signals onboard a vehicle. We also include 5 reference signals from car maneuvers generated with the software `CarSim` (Mechanical Simulation, US). The total time of the maneuvers amounts to 277 s for car and 242 s for helicopter. The bandwith of the inertial signals in the dataset is 10 Hz. The dataset is available online at `https://owncloud.tuebingen.mpg.de/public.php?service=files&t=d81e0425fe350835daa4a92782c5ee24`.

The simulation is performed using the `Simulink` model shown in Fig. 2. The `Controller` block wraps the implementation described in Sec. 3.3 in a `Simulink` C++ S-function. The cable forces calculated by the `Controller` are sent to the `CableRobot` block, which implements the state-space model (2) and calculates the output signal (6).

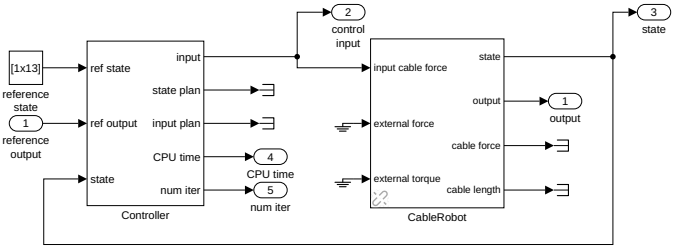

Fig. 2. The core part of the Simulink block diagram used in the simulation.

The state of the plant is sent back to the `Controller` block, which closes the control loop. Time measurement of RTI phases is implemented inside the `Controller` S-function using the `Linux clock_gettime()` function which can provide thread-specific CPU-time clock.

The simulation is run on an `Intel Xeon` CPU @2.67 GHz for each combination of reference trajectory, reference generation strategy, and QP solver. Horizon lengths $N = 2 \dots 60$ are used with `HPMPC` and $N = 2 \dots 30$ with `qpOASES+C`. In order to evaluate reference tracking performance, the RMS of the quantities defined in (10) is computed, aggregated across all reference trajectories.

For computational performance evaluation, the RTI preparation time and feedback time are recorded at every time step for each simulation run. For `HPMPC`, the number of iterations is also recorded. In order to achieve a reliable time measurement, every simulation is repeated 5 times, and the minimum value of CPU time at every iteration is computed between all repetitions. Afterwards, the maximum and average RTI preparation and feedback time, and the maximum and average number of `HPMPC` interior point method iterations are computed by aggregating the data across all reference trajectories.

5. RESULTS

5.1 Tracking performance

Fig. 3 shows how reference tracking performance depends on the horizon length N for the two reference generation strategies and solvers. The QP solutions calculated by the two solvers are almost identical, therefore the tracking performance is identical too. For both reference generation strategies, the specific force RMS error drops rapidly for $N < 40$, and for $N > 40$ it remains almost constant. For long horizons, the "oracle" strategy gives an improvement of a factor of 1.5 in specific force tracking compared to the "constant" strategy. The angular velocity error exhibits a different behavior. For the "constant" strategy, it grows monotonically after $N = 3$, while for the "oracle" strategy it first grows, reaches its maximum at $N = 19$ and decreases afterwards.

The total tracking error for the "constant" strategy converges after $N = 20$. For the "oracle" strategy, it keeps decreasing slowly. The "oracle" strategy gives an improvement in total RMS of a factor 1.1 at $N = 20$ and factor 1.3 at $N = 60$. This gives the upper bound of the improvement that can be achieved by using *any* reference prediction strategy compared to the "constant" strategy for this specific motion simulator and dataset.

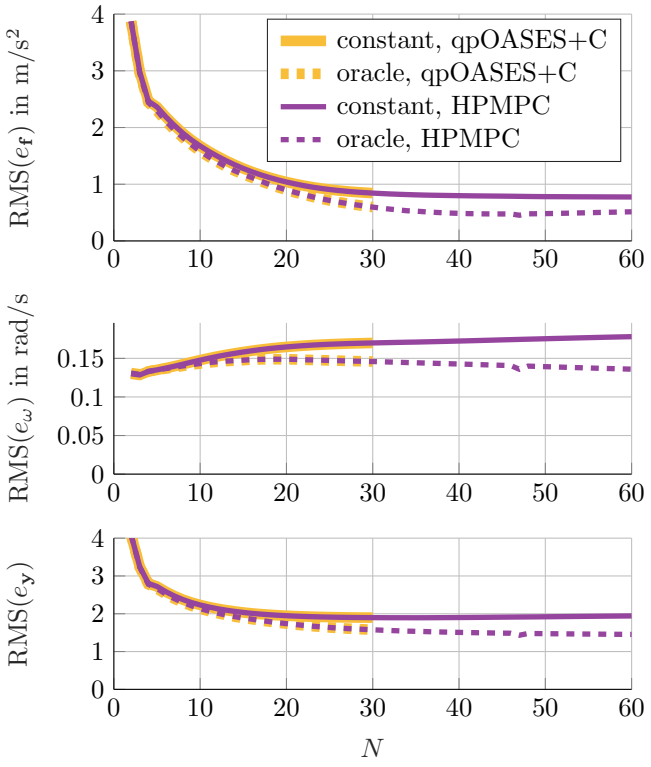

Fig. 3. RMS of error in specific force ($e_\mathbf{f}$), error in angular velocity (e_ω), and total error ($e_\mathbf{y}$) for the two reference generation strategies and two QP solvers depending on horizon length N.

5.2 Computational performance

The CPU time data are presented in Fig. 4. The maximum preparation time increases linearly with N, at a rate of $\approx$ 0.05 ms per time step. The feedback time of the `qpOASES+C` implementation is less than that of `HPMPC` for $N \leq 5$, equal for $N = 5$, and larger (and growing quickly) for $N > 5$, as expected from an algorithm with $\mathcal{O}(N^3)$ complexity. The average feedback time of the `HPMPC` implementation grows linearly with N, and the corresponding maximum feedback time has a peaked pattern with a linear trend of ≈ 0.4 ms per time step.

Fig. 6 shows the average and the maximum number of interior-point method iterations per time step performed by the `HPMPC` solver. While the average number of iterations stays around 8 for the entire range of N, the maximum number of iterations changes irregularly with N between 12 and 70. The peaks in Fig. 4 correspond to the peaks the in number of interior-point method iterations in Fig. 6.

The upper bound of CPU time per MPC step, estimated according to (12), is shown on Fig. 7. We see that for the desired $N = 40$ it can be guaranteed that $T_\text{mpc} \leq T$ if N_iter is restricted to 50.

6. CONCLUSION

Our main finding is that even if the motion platform actuation is included in the MPC model, which increases computational demand, it is still possible to run an MPC-based MCA in real-time with the use of proper software and numerical methods.

In order to achieve a good reference tracking performance with our MPC controller, the horizon length needs to be $N \geq 20$, or ≥ 1 s in terms of time. The feedback time is ≈ 10 times bigger than the preparation time, therefore

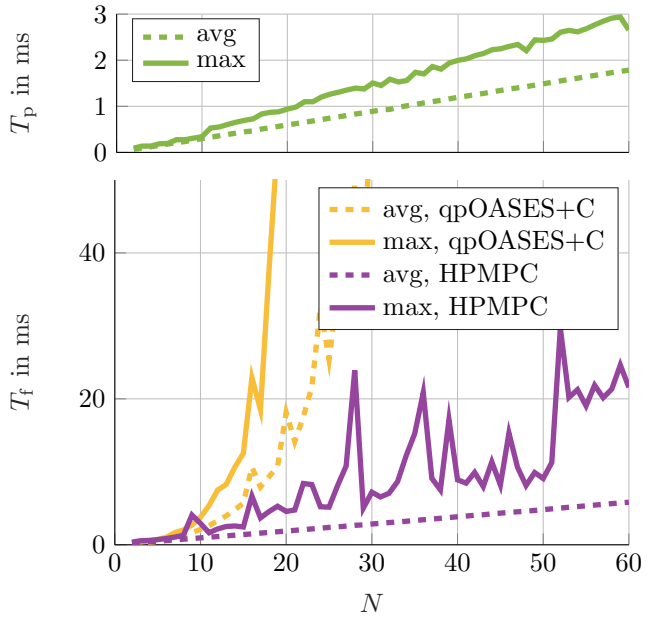

Fig. 4. CPU time of preparation phase T_p and feedback phase T_f depending on horizon length N.

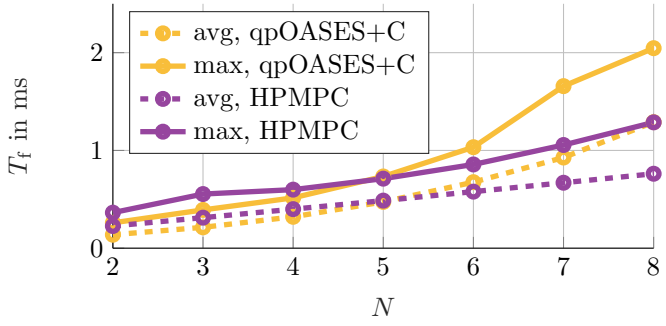

Fig. 5. CPU time of feedback phase (close-up).

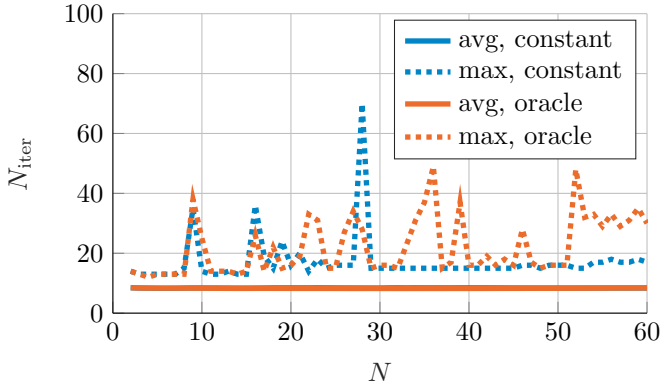

Fig. 6. Number of `HPMPC` interior-point method iterations.

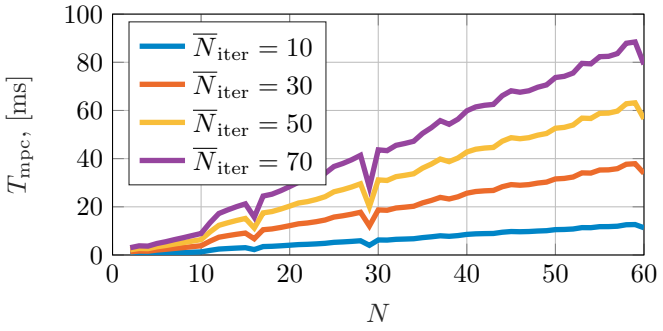

Fig. 7. Max CPU time per MPC step with limited number of interior-point method iterations.

the computational performance is mostly determined by the QP solver. For $N > 5$, `HPMPC` is preferred over `qpOASES+C`. In order to keep computation time T_{mpc} below the sampling time T, the number of `HPMPC` interior-point method iterations must be restricted to 50.

Increasing the horizon length beyond $N = 40$ does not change the tracking performance for the "constant" reference generation strategy. Employing a reference prediction algorithm can reduce the reference tracking error by a factor of 1.1 to 1.3. The improvement is higher for longer horizons.

The final goal is to implement the MPC controller on the real motion simulator. As an intermediate step, the controller will first be tested on a scaled down version of the simulator.

ACKNOWLEDGEMENTS

This research was supported by the EU via ERC-HIGHWIND (259 166), ITN-TEMPO (607 957), and ITN-AWESCO (642 682) and by DFG in context of the Research Unit FOR 2401.

REFERENCES

Andersson, J. (2013). *A General-Purpose Software Framework for Dynamic Optimization.* PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

Beghi, A., Bruschetta, M., and Maran, F. (2012). A real time implementation of MPC based Motion Cueing strategy for driving simulators. In *Proceedings of the IEEE Conference on Decision and Control*, 6340–6345. doi:10.1109/CDC.2012.6426119.

Dagdelen, M., Reymond, G., Kemeny, A., Bordier, M., and Maïzi, N. (2009). Model-based predictive motion cueing strategy for vehicle driving simulators. *Control Engineering Practice*, 17(9), 995–1003. doi:10.1016/j.conengprac.2009.03.002.

Diehl, M. (2001). *Real-Time Optimization for Large Scale Nonlinear Processes.* Ph.D. thesis, Universität Heidelberg.

Domahidi, A., Zgraggen, A., Zeilinger, M., Morari, M., and Jones, C. (2012). Efficient interior point methods for multistage problems arising in receding horizon control. In *IEEE Conference on Decision and Control (CDC)*, 668 – 674. Maui, HI, USA.

Fang, Z. and Kemeny, A. (2012). Explicit MPC motion cueing algorithm for real-time driving simulator. In *Conference Proceedings - 2012 IEEE 7th International Power Electronics and Motion Control Conference - ECCE Asia, IPEMC 2012*, volume 2, 874–878. doi:10.1109/IPEMC.2012.6258965.

Ferreau, H., Kirches, C., Potschka, A., Bock, H., and Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4), 327–363.

Frasch, J.V., Sager, S., and Diehl, M. (2013). A Parallel Quadratic Programming Method for Dynamic Optimization Problems. *Mathematical Programming Computation.* (submitted).

Frison, G., Sørensen, H.B., Dammann, B., and Jørgensen, J.B. (2014). High-performance small-scale solvers for linear model predictive control. In *Control Conference (ECC), 2014 European*, 128–133. IEEE.

Garrett, N.J.I. and Best, M.C. (2010). Driving simulator motion cueing algorithmsa survey of the state of the art. In *Proceedings of the 10th International Symposium on Advanced Vehicle Control (AVEC)*, 183–188.

Katliar, M., de Winkel, K.N., Venrooij, J., Pretto, P., and Bülthoff, H.H. (2015). Impact of MPC Prediction Horizon on Motion Cueing Fidelity. In *Driving Simulation Conference & Exhibition 2015*, 219–222.

Kouzoupis, D., Zanelli, A., Peyrl, H., and Ferreau, H.J. (2015). Towards proper assessment of QP algorithms for embedded model predictive control. In *Proceedings of the European Control Conference (ECC).*

Miermeister, P., Lächele, M., Boss, R., Masone, C., Schenk, C., Tesch, J., Kerger, M., Teufel, H., Pott, A., and Bülthoff, H.H. (2016). The cablerobot simulator large scale motion platform based on cable robot technology. *IROS.*

Nahon, M.A., Reid, L.D., and Kirdeikis, J. (1992). Adaptive simulator motion software with supervisory control. *Journal of Guidance, Control, and Dynamics*, 15(2), 376–383. doi:10.2514/3.20846.

Nahon, M.A. and Reid, L.D. (1990). Simulator motion-drive algorithms - A designer's perspective. *Journal of Guidance, Control, and Dynamics*, 13(2), 356–362. doi:10.2514/3.20557.

Rawlings, J. and Mayne, D. (2009). *Model Predictive Control: Theory and Design.* Nob Hill Pub.

Schmidt, S.F. and Conrad, B. (1970). Motion drive signals for piloted flight simulators. Technical Report NASA-CR-1601, NASA, Washington, United States.

Sivan, R., Ish-Shalom, J., and Huang, J. (1982). An Optimal Control Approach to the Design of Moving Flight Simulators. *IEEE Transactions on Systems, Man and Cybernetics*, 12(6), 818–827. doi:10.1109/TSMC.1982.4308915.

Telban, R.J. and Cardullo, F. (2005). Motion cueing algorithm development: Human-centered linear and nonlinear approaches. *NASA TechReport*, (May), CR–2005–213747.

Venrooij, J. (2014). VIA-Sync: A novel vehicle motion recording platform for synchronised visuo-inertial playback. In *Vehicle Dynamics Conference 2014.*